Gilded Rose Refactoring Assignment

Tharanitharan Muthuthirumaran

Problems:

1. One of the biggest problems with the current program is it is deeply nested. Currently, it is having nearly 6+ levels of if blocks. It is very error prone and there are some duplicate checks.
2. The conditions have hardcoded strings which are very prone to errors.
3. It does not follow the DRY (Don't Repeat Yourself) and violates the Single Responsibility Principle.
4. It is very poort testability and there is no clear structure for the program.
5. There are no comments in the code.

UML Diagram:

Added UML diagram as a PDF.

Design Explanation

I used the Strategy Pattern, which can be found on the refactoring.guru website, to split each of the item types logic into separate classes. So, we have NormalItem class, AgedBrie class, BackstagePass class, and so on. This has greatly cleaned up our code, as each class only deals with one type of item. I also used the Factory Method Pattern, which creates a class called StrategyFactory that decides which strategy to use, depending on the name of the item. This way, when we need to implement the Conjured items, I can create a small class called Conjured without touching any of the existing code. The ItemUpdateStrategy class has common helper methods, such as `_clamp_quality()`, which prevents code duplication in all of the classes. We went from a single complex method to separate small classes each, which are much easier to understand and test separately.